#include<iostream> #define infinity 9999 #define MAX 20 #include<stdlib.h>

using namespace std; typedef struct edg { int s,d,w; }edg;

class graph {

int G[MAX][MAX]; int n; int set[20][20],cnt[20]; int noe; edg e[100];

public: graph(){

n=0;

}

void readgraph();

void printgraph();

void edgelist();

void selection\_sort(edg e[100],int n);

int findset(int x);

int Kruskal(graph &mst);

};

void graph::readgraph()

{

int i,j,start\_vertex,end\_vertex,weight;

cout<<"\nEnter number of vertices:\t"; //accepting vertices and edges

cin>>n;

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

G[i][j]=0; // initialising the array

}

cout<<"Enter number of Edges:\t";

cin>>noe;

for(i=0;i<noe;i++)

{

cout<<"Enter an Edge (start,end,Weight) :";cin>>start\_vertex>>end\_vertex>>weight;

G[start\_vertex][end\_vertex]=G[end\_vertex][start\_vertex]=weight;

}

}

void graph::printgraph()

{

int i,j;

cout<<"\nList of edges in the spanning tree:";

for(i=0;i<n-1;i++)

{

for(j=i+1;j<n;j++)

if(G[i][j]!=0) // printing shortest path

cout<<"\n(start,end,weight) => ("<<i<<","<<j<<","<<G[i][j]<<")";

}

}

void graph::edgelist()

{

int i,j,k=0;

for(i=0;i<n;i++)

{

for(j=i;j<n;j++)

{

if(G[i][j]!=0)

{

e[k].s=i;e[k].d=j;

e[k].w=G[i][j];

k++;

}

}

}

cout<<"source destination weight\n";

for(i=0;i<k;i++)

{

cout<<e[i].s<<"\t";

cout<<e[i].d<<"\t";

cout<<e[i].w<<"\n";

}

selection\_sort(e,k);

}

void graph::selection\_sort(edg e[100],int n)

{

int temp=0,min=0,j,pos=0,k,i,mins=0,mind=0,temps=0,tempd=0;

edg tempe,mine;

for(i=0;i<n;i++)

{

mine=e[i];pos=i;

for(j=i+1;j<n;j++)

{if(e[j].w<mine.w)

{

mine=e[j];

pos=j;

}

}

tempe=e[i];

e[i]=mine;

e[pos]=tempe;

}

cout<<"\n sorted elements are";

for(k=0;k<n;k++)

{

cout<<e[k].s<<"\t";

cout<<e[k].d<<"\t";

cout<<e[k].w<<"\n";

}

}

int graph::findset(int x)

{

int i,j;

for (i=0;i<n;i++)

{

for (j=0;j<cnt[i];j++){

if(x==set[i][j])

return i;

}

}

return -1;

}

int graph::Kruskal(graph &mst)

{

int i,j,min\_cost=0;

edgelist();

for (i=0;i<n;i++)

{

set[i][0]=i;

cnt[i]=1;

}

cout<<"\n List of New Edges is:\n";

for (i=0;i<noe;i++)

{

int p1=findset(e[i].s);

int p2=findset(e[i].d);

if(p1!=p2)

{

mst.G[e[i].s][e[i].d]=e[i].w;mst.G[e[i].d][e[i].s]=e[i].w;

min\_cost=min\_cost+e[i].w;

cout<<e[i].s<<"\t"<<e[i].d<<"\t"<<e[i].w<<"\n";

for(j=0;j<cnt[p2];j++)

{

set[p1][cnt[p1]]=set[p2][j];

cnt[p1]++;

}

cnt[p2]=0;

}

}

return min\_cost;

}

int main()

{

int opt,total\_cost;

graph g,mst;

do

{

cout<<"\n\n1.Read the Graph\n2.Kruskal Algorithm \nEnter your choice :";

cin>>opt;

switch(opt)

{case 1:g.readgraph();

g.printgraph();

break;

case 2:

total\_cost=g.Kruskal(mst);

cout<<"\n Min Cost is = "<<total\_cost<;

break;

}

}while(opt!=3);

}